VLSI Architectures for the Multiplication of Integers Modulo a Fermat Number

J. J. Chang
Electronic Parts Reliability Section

T. K. Truong
Communications Systems Research Section

I. S. Reed and I. S. Hsu
University of Southern California

Multiplication is central in the implementation of Fermat number transforms and other residue number algorithms. There is need for a good multiplication algorithm that can be realized easily on a VLSI chip. In this article, the Leibowitz multiplier is modified to realize multiplication in the ring of integers modulo a Fermat number. This new algorithm requires only a sequence of cyclic shifts and additions. The designs developed for this new multiplier are regular, simple, expandable, and, therefore, suitable for VLSI implementation.

I. Introduction

Fermat number transforms (FNTs) were developed to compute cyclic convolutions (Refs. 1 and 2) and to encode and decode a (256, 224) Reed-Solomon (R-S) code in $GF(F_3)$, where $F_3 = 2^{2^3} + 1 = 257$ (Ref. 3). Recently Leibowitz (Ref. 4) proposed the diminished -1 representation for binary arithmetic operations modulo F_t. Based on the ideas in Ref. 4, a new algorithm is developed to realize a multiplier over the ring of integers modulo a Fermat number. This algorithm requires only cyclic shifts and additions. An example, illustrating both the pipeline and systolic array aspects of this structure, is given for a multiplier in the field of $GF(2^{2^2} + 1)$.

II. Multiplication Modulo F_t

In this section, a new algorithm is developed for multiplication in the ring of integers modulo $F_t = 2^{2^t} + 1$. This new algorithm is illustrated by the example for $t = 2$. The same structure clearly extends to more general multiplication algorithms over F_t.

To perform efficiently, the binary arithmetic operations modulo F_t, the diminished -1 representation proposed by Leibowitz (Ref. 4), are used. Table 1 shows the correspondence of elements in $GF(2^{2^2} + 1)$ with their decimal equivalents in a normal binary representation and with their values in the diminished -1 representation. In the Leibowitz diminished -1 representation, the most significant bit (MSB) can be viewed as the zero-detection bit.

To realize this new algorithm for multiplication, let A, B be two binary integers and $A - 1, B - 1$ their diminished -1 representations. It is now desired to perform a multiplication of the two positive numbers $A - 1$ and $B - 1$ in their diminished -1 representation. Note that if the MSB of either $A - 1$ or $B - 1$ is one, then the multiplication is inhibited, and the product is zero.

To accomplish this new multiplication process, one first translates $B - 1$ from the diminished -1 representation to the normal binary representation, B. Then the multiplication of $A - 1$ and B is performed as follows:

$$(A-1)\cdot B = (A-1)\left(\sum_{k=0}^{4} b_k 2^k\right) = \sum_{k=0}^{4} b_k 2^k \cdot (A-1) \tag{1}$$

Multiplying the diminished -1 numbers by the different powers of two in Eq. (1) so that the result is a diminished -1 number. This is achieved by the identity $2^k(A-1) + (2^k - 1) = 2^k A - 1$ for $0 \leqslant k \leqslant 4$. Thus when considered as a sum of diminished numbers, Eq. (1) becomes

$$(A-1)B = \sum_{k=0}^{4} b_k \cdot (2^k A - 1) = \sum_{k=0}^{4} (d_k - 1) \tag{2}$$

where $d_k - 1 = b_k \cdot (2^k A - 1)$ for $0 \leqslant k \leqslant 4$.

For $d_k - 1$ with $0 \leqslant k \leqslant 4$ to be diminished -1 numbers, they must add correctly pairwise in accordance with the formula,

$$(d_i - 1) + (d_j - 1) + 1 = (d_i + d_j - 1) \tag{3}$$

It is readily verified that this holds in all four cases of (b_i, b_j). Hence the diminished -1 addition in Eq. (2) can be performed recursively in the following manner:

$$(A-1)\cdot B = ((((d_0 + d_1 - 1) + d_2 - 1) + d_3 - 1) + d_4 - 1) \tag{4}$$

A substitution of Eq. (3) into Eq. (4) yields

$$\begin{aligned}(A-1)\cdot B = {} & (((((d_0 - 1) + (d_1 - 1) + 1) \\ & + (d_2 - 1) + 1) + (d_3 - 1) + 1) + (d_4 - 1) + 1)\end{aligned} \tag{5}$$

where $d_k - 1 = b_k(2^k A - 1)$.

From Eq. (5), the following identity is obtained for the product $A \cdot B$ diminished by 1:

$$\begin{aligned}(A-1)\cdot B &= b_0 \cdot (A-1) + b_1 \cdot (2A-1) \\ &\quad + b_2 \cdot (2^2 A - 1) + b_3 \cdot (2^3 A - 1) \\ &\quad + b_4 \cdot (2^4 A - 1) + 4 \\ &= A \cdot B - S + 4 = (A \cdot B - 1) - S + 5\end{aligned} \tag{6}$$

where $S = b_0 + b_1 + b_2 + b_3 + b_4$.

It follows from Eq. (6) that $A \cdot B$ in diminished -1 notation is

$$(A \cdot B - 1) = (A-1)\cdot B - (4 - S) - 2 + 1 \tag{7}$$

Now let

$$D = 4 - S \tag{8}$$

where $0 \leqslant D \leqslant 4$. Since $2^4 + 1 \equiv 0 \bmod F_2$, one has $-D - 2 = 2^4 + 1 - D - 2 = (2^4 - D - 1) = \overline{D}$, the binary one's complement of D. Hence, by Eq. (7), the diminished -1 representation of $A \cdot B$ becomes

$$C = (A \cdot B - 1) = (A-1)\cdot B + \overline{D} + 1 \tag{9}$$

where $\overline{D}$ is the one's complement of D in Eq. (8). A substitution of Eq. (4) into Eq. (9) yields

$$\begin{aligned}(A \cdot B - 1) = {} & (((((\overline{D} + b_0 \cdot (2^0 A - 1) + 1) + b_1 \cdot (2^1 A - 1) \\ & + 1) + b_2 \cdot (2^2 A - 1) + 1) + b_3 \cdot (2^3 A - 1) \\ & + 1) + b_4 \cdot (2^4 A - 1) + 1)\end{aligned} \tag{10}$$

as the new multiplication algorithm.

Let $Co = \overline{D}$. Then the multiplication algorithm in Eq. (10) can be put into the following recursive form:

$$C_{k+1} = C_k + b_k \cdot (2^{k+1} \cdot A - 1) + 1 \text{ for } 0 \leqslant k \leqslant 4 \tag{11a}$$

If one successively computes C_{k+1} in Eq. (11a) for $0 \leqslant k \leqslant 4$, then the required result is obtained as follows:

$$C_5 = C_4 + b_4(2^4A - 1) + 1 = (A \cdot B - 1) = C \tag{11b}$$

Example 1: A recursive diminished -1 multiplication algorithm: Let $A - 1 = 0\,1\,0\,1\,0$, $B - 1 = 0\,0\,1\,0\,1$; compute $C = (A \cdot B - 1) = 0\,1\,0\,1\,0 \times 0\,0\,1\,0\,1$ modulo $2^4 + 1$.

To compute C, one first translates $B - 1$ to B. That is, $B = B - 1 + 1 = 0\,0\,1\,0\,1 + 1 = 0\,0\,1\,1\,0 = b_4\,b_3\,b_2\,b_1\,b_0$. From Eq. (10), the sequence of computations for $0\,1\,0\,1\,0 \times 0\,0\,1\,0\,1$ is then as follows:

$$\begin{array}{rl}
01101 & C_0 = \bar{D} = 01101 \\
+00000 & b_0 \cdot (2^0A - 1) = 0(01010) = 00000 \\
\hline
01101 & \\
+ \searrow 1 & \\
\hline
01110 & C_1 = C_0 + b_0 \cdot (2^0A - 1) + 1 \\
+00100 & b_1 \cdot (2^1A - 1) = 1 \cdot (00100) = 00100 \\
\hline
10010 & \\
+ \searrow 0 & \\
\hline
00010 & C_2 = C_1 + b_1 \cdot (2^1A - 1) + 1 \\
+01001 & b_2 \cdot (2^2A - 1) = 1 \cdot (01001) = 01001 \\
\hline
01011 & \\
+ \searrow 1 & \\
\hline
01100 & C_3 = C_2 + b_2 \cdot (2^2A - 1) + 1 \\
+00000 & b_3 \cdot (2^3A - 1) = 0 \cdot (00010) = 00000 \\
\hline
01100 & \\
+ \searrow 1 & \\
\hline
01101 & C_4 = C_3 + b_3 \cdot (2^3A - 1) + 1 \\
+00000 & b_4 \cdot (2^4A - 1) = 0 \cdot (00101) = 00000 \\
\hline
01101 & \\
\searrow 1 & \\
\hline
01110 & C_5 = C_4 + b_4(2^4A - 1) + 1 = C
\end{array}$$

Thus $C = 0\,1\,1\,1\,0$ is the desired result of $0\,1\,0\,1\,0$ times $0\,0\,1\,0\,1$, modulo $2^4 + 1$ in diminished -1 notation.

III. A VLSI Structure for Implementing Multiplication Modulo F_t

Example 1 of the new diminished -1 multiplication algorithm in the previous section shows that diminished -1 additions require the addition of the complement of an end around carry to its sum. A considerable speed improvement can be obtained by performing this operation simultaneously with the summation. A modified algorithm with this simultaneous addition is given for the previous example as follows:

Example 2: Modified recursive diminished -1 multiplication:

$$\begin{array}{rl}
11101 & C_0 = 10000 + \bar{D} = 11101 \\
00000 & b_0 \cdot (2^0A - 1) = 0(01010) = 00000 \\
+ \searrow 0 & \\
\hline
01101 & C_1 = C_0 + b_0 \cdot (2^0A - 1) + 1 \\
00100 & b_1(2^1A - 1) = 1 \cdot (00100) = 00100 \\
+ \searrow 1 & \\
\hline
10010 & C_2 = C_1 + b_1(2^1A - 1) + 1 \\
01001 & b_2(2^2A - 1) = 1 \cdot (01001) = 01001 \\
+ \searrow 0 & \\
\hline
01011 & C_3 = C_2 + b_2(2^2A - 1) + 1 \\
00000 & b_3(2^3A - 1) = 0 \cdot (00010) = 00000 \\
+ \searrow 1 & \\
\hline
01100 & C_4 = C_3 + b_3(2^3A - 1) + 1 \\
00000 & b_4(2^4A - 1) = 0 \cdot (00101) = 00000 \\
+ \searrow 1 & \\
\hline
01101 & C_5 = C_4 + b_4(2^4A - 1) + 1 \\
00000 & 0(2^5A - 1) = 0 \cdot (01011) = 00000 \\
+ \searrow 1 & \\
\hline
01110 & C = C_5 + 1
\end{array}$$

A possible VLSI structure for Example 2 is presented in Fig. 1. In Fig. 1, A, B, and C are 4-bit, 6-bit, and 5-bit registers, respectively. Initially, registers A, B, and C contain the multiplicand in the diminished -1 representation, the multiplier in normal representation, and $2^4 + D$, respectively. At the very same moment, $C_{k+1} = C_k + b_k(2^kA - 1)$ is computed and loaded into the C register. Simultaneously, the diminished -1 multiplication of $(A - 1)$ by 2 is performed first by a left cyclic shift of the four least-significant bits of the register A with the A_3-bit circulated into the first significant bit complemented. Also, at the same time, register B is shifted right by one bit. These operations are continued repetitively until the MSB of the register B is shifted out. The desired final result of 0 1 1 1 0 after 5 iterations is obtained in register C.

The layout of the structure in Fig. 1 has been completed with the use of the CAESAR design tool (Ref. 5). The final layout of the multiplication chip is shown in Fig. 2.

In the future, logic and circuit simulations will be performed. These will be followed by chip fabrication and final testing. The total number of transistors in this chip is about 300. The area of the chip is estimated to be about 12,000 λ^2.

References

1. Rader, C. M., "Discrete Convolution via Mersenne Transforms," *IEEE Trans. Computers*, Vol. C-21, No. 12, pp. 1269-1273, Dec. 1972.

2. Agarwal, R. C., and C. S. Burns, "Fast Convolution Using Fermat Number Transforms with Applications to Digital Filtering," *IEEE Trans. Acoustics, Speech and Signal Processing*, Vol. ASSP-22, No. 2, pp. 87-97, April 1974.

3. Reed I. S., T. K. Truong, and L. R. Welch, "The Fast Decoding of Reed-Solomon Code Using Fermat Number Transforms," *IEEE Trans. Information Theory*, Vol. IT-24, No. 4, pp. 497-499, July 1978.

4. Leibowitz, L. M., "A Simplified Binary Arithmetic for the Fermat Number Transform," *IEEE Trans., Acoustic, Speech and Signal Processing*, Vol. ASSP-24, No. 5, pp. 356-359, Oct. 1976.

5. Ousterhout, J., *Editing VLSI Circuits with Caesar*, Computer Science Division, Electrical Engineering and Computer Sciences, University of California, Berkeley, April 21, 1982.

Table 1. The correspondence among decimal numbers, their values in the normal binary representation, and in the diminished −1 representation

Decimal number	Normal binary representation	Diminished −1 representation
0	0 0 0 0 0	1
1	0 0 0 0 1	2
2	0 0 0 1 0	3
3	0 0 0 1 1	4
4	0 0 1 0 0	5
5	0 0 1 0 1	6
6	0 0 1 1 0	7
7	0 0 1 1 1	8
8	0 1 0 0 0	9 (−8)
9 (−8)	0 1 0 0 1	10 (−7)
10 (−7)	0 1 0 1 0	11 (−6)
11 (−6)	0 1 0 1 1	12 (−5)
12 (−5)	0 1 1 0 0	13 (−4)
13 (−4)	0 1 1 0 1	14 (−3)
14 (−3)	0 1 1 1 0	15 (−2)
15 (−2)	0 1 1 1 1	16 (−1)
16 (−1)	1 0 0 0 0	0

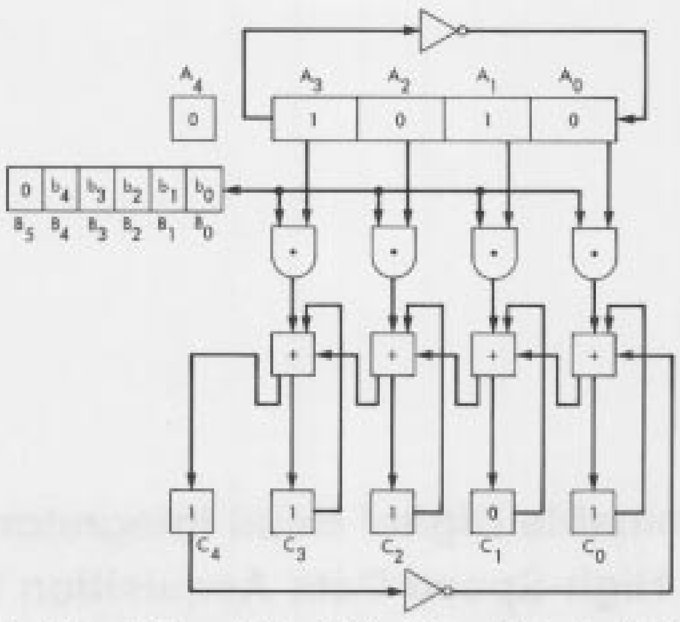

Fig. 1. The pipeline architecture for the implementation of multiplication modulo the Fermat number $2^4 + 1$ using diminished -1 number representations

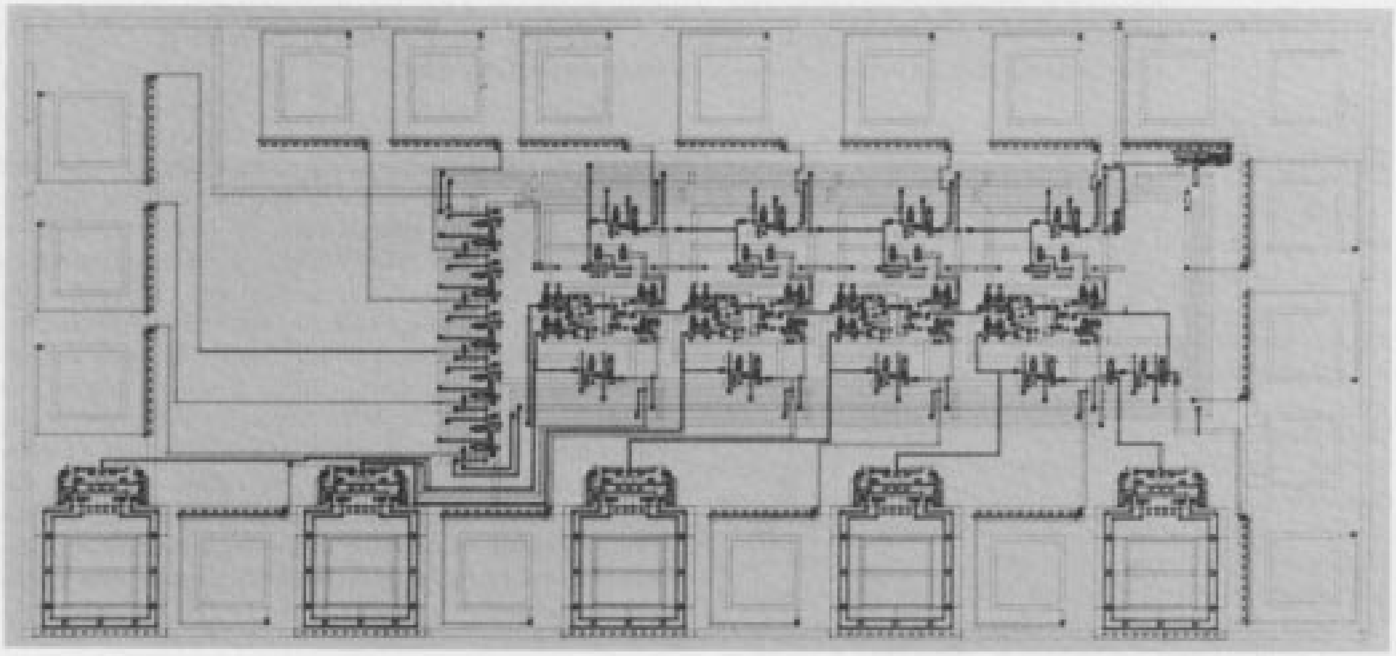

Fig. 2. The VLSI nmos technology layout for multiplication modulo the Fermat number $2^4 + 1$, using diminished -1 number representation